Maastricht University

Faculty of Psychology and Neuroscience

Multi-Source Domain Adaptation:
A Representational Transfer from Natural Images to Clip Art and Sketches

Bachelor thesis

Marius Marten Kästingschäfer

Student-Id: i6128643

20.05.2019, Maastricht, The Netherlands

Multi-Source Domain Adaptation: A Representational Transfer from Natural Images to Clip Art and Sketches

Marius Marten Kästingschäfer
Faculty of Psychology and Neuroscience, Maastricht University, Maastricht, The Netherlands

Humans are able to recognize scenes independently of the modality they perceive it in. In this paper, it is tested if scene specific features extracted from natural images are reusable for classifying clip art and sketches. Further, the ability of networks to hold multiple representations simultaneously is examined. To study this problem a pretrained convolutional neural network is used. Further, a randomly initialized classifier is added and retrained. The representations of clip art achieve the highest results, followed by sketches. Natural image performance remains notably below prior results. The experiment suggests that the overall transferability of learned features is not only limited by the distance but also by the diversity of the input distribution. Moreover, a multimodal representation is shown to be feasible but only poorly. Further research is needed to explain the obtained outcomes and to explain why they deviate from previous expectations.

Keywords: deep learning, transfer learning, computer vision, cross-domain learning, multi-modal, fine-tuning, scene recognition

Introduction

Image recognition, the task to identify and detect items like places, buildings or people in images, has been one of the most intensively studied topics within the machine learning community in the last years. In 2012 the AlexNet (Krizhevsky, Sutskever, & Hinton, 2012) won the ImageNet Large Scale Visual Recognition Challenge with an implementation of a convolutional neural network (ConvNet) and made the technique the state-of-the-art for image classification. Since then, ConvNets are the superior approach for nearly all recognition and detection tasks (Szegedy et al., 2015). These accomplishments were mainly attained on natural images and are now getting closer to human performance (Geirhos et al., 2017). However, these achievements remain difficult to replicate across different modalities. In this context both, modalities and domains are defined as information from multiple visual sources. Human adults can easily detect classes, such as a kitchen, across domains (DiCarlo & Cox, 2007), whereas ConvNets equipped with generic representations often don't possess this capability (Chollet, 2017; Azizpour, Razavian, Sullivan, Maki, & Carlsson, 2014). Generic since those representations are acquired by changing the network's parameters during processing a predefined set of images, not through interaction with the physical environment. *Figure 1* shows an example of some concepts represented in different modalities. Classifying these scenes is possible with minimal effort for most people. The problem for computer vision models is that modalities such as sketches, drawings and clip art often lack the availability of millions of annotated pictures in curated datasets. Limited amounts of images yield the problem that the model is only able to classify the training images and no new ones. This issue is called overfitting. The model then contains more parameters than justified by the data (Bishop, 2006). This makes deep convolutional neural networks with multiple layers and millions of parameters in cases of limited data especially prone to overfitting. Therefore deep networks become unable to grasp the right features underlying the image domain. Transfer learning tries to overcome the problem of scarce resources by using a certain source domain to learn a classifier for unseen data in a target domain. Good reusability of models has numerous advantages, namely a decrease in training time, being often less computationally expensive, and the need for less labelled data than training a model from scratch. Due to the vast field of transfer learning this paper will focus on transductive domain adaptation applied to visual tasks. During transductive domain adaptation (sometimes also called transfer learning) the tasks remain the same across domains but the domains themself differ. A major difficulty in domain adaptation is how to adapt the input distribution of image sets effectively. It will be tested whether the assumption underlying domain adaptation, namely that input distributions are shared between two domains and as such have detectable features in common, holds. To study this problem, a ConvNet trained on natural images will be reused to classify unfamiliar domains. In particular an image classification architecture optimized for the Places365 set will be fine-tuned on the target domain and used to classify clip art and sketches with identical scene labels. This way it will also be tested to what extent the model can represent mul-

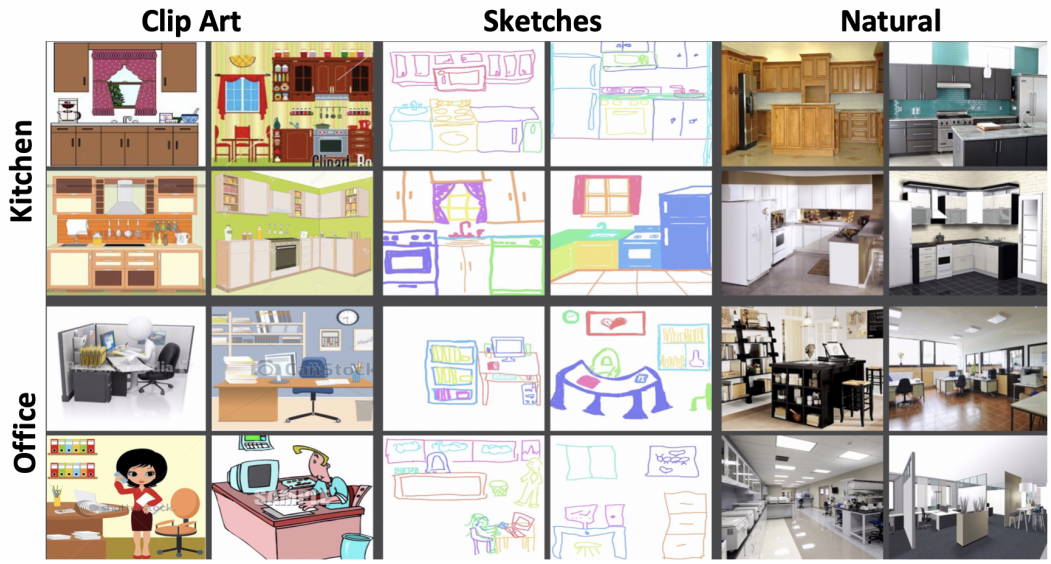


Figure 1. **Can you recognize scenes across different styles?** The figure shows a kitchen and an office (concepts) across different modalities (clip art, sketches and natural images).

tiple modalities in one network. This approach builds on prior research done in the field of domain adaptation (Csurka, 2017) and multimodal learning methods (Aytar, Castrejon, Vondrick, Pirsiavash, & Torralba, 2018) in computer vision. The exact research question followed by hypotheses is given in the next section. The remaining parts of the paper are organized as follows. In the next section, related work and a biological inspiration for deep learning to improve computer vision is introduced. Then, a description and insights into the proposed method is provided. Afterwards, the experimental setup is described. Finally, the results are presented and discussed, followed by a conclusion.

Research questions and Hypotheses

Despite recent interest to decipher how the final output from machine learning algorithms is derived, deep models often remain black boxes. The following questions join this line of investigations, focusing on abilities and limitations of representing multiple domains. Whether the three modalities possess different input distributions lead to the first question. How well does the chosen image classification architecture trained on natural images perform on classifying clip art and drawn images? The second question targets the transferability of representations in the convolutional base by adjusting the network. Namely, are the features extracted from a wide range of natural images reusable for classifying clip art and sketches?

The hypotheses are built on assessing the literature available in the field and some prior test experience. First it is assumed that the image classification architecture will only be able to classify some images from the new domain. Secondly, since the features in the utilized ConvNet were extracted from a broad range of scenes, it is assumed that they are general enough to be adjusted to modalities they were not trained on. This hypothesis implicitly assumes that spatial patterns in clip art and drawings are not fundamentally different from the ones in natural images.

Transfer learning

Like almost every field in deep learning, the field of transfer learning has grown during the last years and has made tremendous progress. To get a comprehensive overview of the topic, transfer learning and related research is described. Focusing on visual tasks, a comprehensive explanation of transfer learning is given, the transferability of features is investigated and some related methods are mentioned.

What is transfer learning? To understand transfer learning it is important to explain source and target data. The term source data describes the data the network is originally trained on, in case of the network used in this paper the source data consists of natural images. The target data refers to the data the network should be adapted to, for the purpose of this experiment those are clip art and drawings. Transfer learning, unlike normal machine learning, aims at training a model on source data that is different from the target data. Compared to classical machine learning transfer learning achieves better results in domains with limited amounts of training data (Coutinho, Deng, & Schuller, 2014). The use of previously learned features makes transfer learning also beneficial in terms of efficiency. When transfer learning is done by supervision, labelled data is used to fine-tune existing feature representations. Historically, knowledge trans-

ferring was mainly done within the same modalities (Aytar & Zisserman, 2011; Ganin & Lempitsky, 2014) and some first steps towards multimodal learning were taken by generating a shared representation across modalities (Ngiam et al., 2011). Transfer learning proves to be well suited for close domain shifts (Luo, Zheng, Guan, Yu, & Yang, 2018). It is important that the source and the target domain are from similar domains, and that the task is similar across those. Traditional machine learning is characterized by using equal source and target domains and by performing the same task on both. During transductive transfer learning (sometimes also called domain adaptation) the tasks remain the same across domains but the domains themself differ. Datasets for domain adaptation consist of different modalities such as natural images, line drawings or cartoons. Since those images have no one-to-one correspondence between modalities they are called weakly aligned. An example of strong alignment would be a drawing based on a natural image. The alignment of modalities is important because it determines the transferability of features.

What can be transferred? To reuse previously learned features they need certain properties. Those properties are called transferability characteristics or transferability factors (Azizpour et al., 2014). Transferability factors define what part of knowledge can be transferred and how it can be transferred best. To assess the transferability of features, one has to look at the characteristics of the underlying dataset. B. Zhou, Lapedriza, Xiao, Torralba, and Oliva (2014) chose diversity (number of classes) and density (number of images per class) to be such characteristics. All images taken in one kitchen, would have a low diversity but a high density. Measurements of the Places205 showed a high diversity of the dataset compared with the ImageNet and the SUN dataset (B. Zhou et al., 2014). Also, the Places205 is a scene-centric dataset. The difference between object-centric and scene-centric images plays a role because iconic pictures of objects don't contain the richness of visual information that images of scenes provide (B. Zhou et al., 2014). According to Donahue et al. (2013) the reason for this is that for scene recognition a grasp of the entire image is needed. Donahue et al. (2013) also showed that even within scene-centric images, features trained on indoor and outdoor scenes, differ significantly. Resulting in the first rule of thumb: scene-centric images should be transferred to data containing a lower or the same richness. This is a criterion the approach in this paper follows. How to transfer best depends on finding the hyperparameter that maximizes transferability. Unfortunately, there is often no one single solution that suits every problem equally well. What could be generalized is what Azizpour et al. (2014) showed, that interrupting training (early stopping) does often not improve transferability of features. What remains to be true for all transfer task is that an increasing amount of target data improves performance significantly (Soekhoe, van der Putten, & Plaat, 2016). The vast amount of literature can be summarized in the second rule of thumb: With increasing distance between tasks the transferability gap grows. The same holds true for an increasing space between input distributions of the source and target dataset (Yosinski, Clune, Bengio, & Lipson, 2014). An extreme example for input distributions that might lie too far apart for transfer learning is sentiment analysis where one tries to transfer knowledge from acoustic data to classify the sentiment of images. Whether the input distributions are too far apart between natural images, clip art and drawings can only be answered after the experiment.

Which methods are related? Related techniques are increasing in number during the last years. For the purpose of this paper it will be enough to only briefly mention two related techniques. First, multimodal deep learning combines complementary information from multiple modalities to improve performance. Multi-modal learning is called multi-task learning when multiple learning tasks are solved at the simultaneously (Pan & Yang, 2010). Complications could arise due to varying levels of noise in the data and evident conflicts between modalities. Second, One-Shot/Zero-Shot Learning trains a classifier by using only a very limited number of examples. This method is useful to exploit datasets in which the costs of labeling data is high or unfeasible. Interested readers can consult Socher et al. (2013); Wang, Zheng, Yu, and Miao (2019) or Fu et al. (2017).

Biological inspiration

Animals and especially Homo sapiens are able to leverage knowledge and experiences independently of the modality they perceive it in (Milne, Wilson, & Christiansen, 2018; DiCarlo & Cox, 2007). A similar capability in machines would be an outstanding achievement. This is one of the many reasons why the use of neuroscientific and biological models was and still is an important source of inspiration for some areas of research (Hassabis, Kumaran, Summerfield, & Botvinick, 2017; Lake, Ullman, Tenenbaum, & Gershman, 2016). For example the idea that networks of neurons might learn via using supervisory feedback was introduced by the psychologist Rosenblatt (1958). Some years later Hubel and Wiesel (1962) made single-cell recordings from the mammalian visual cortex, that inspired filter and pooling methods in convolutional neural networks (Sejnowski, 2018). Moreover, current network architectures imitate the hierarchical organization of cortical sensory systems with a successive, nested information flow (Riesenhuber & Poggio, 1999; Serre, Wolf, Bileschi, Riesenhuber, & Poggio, 2007). Even with some cortical-cortical projections in the brain being inconsistent with pure hierarchical processing, the hierarchical structure is relevant nevertheless (Hawkins, Lewis, Klukas, Purdy, & Ahmad, 2018; Markov et al., 2014).

In recent years Yamins et al. (2014) showed how similar

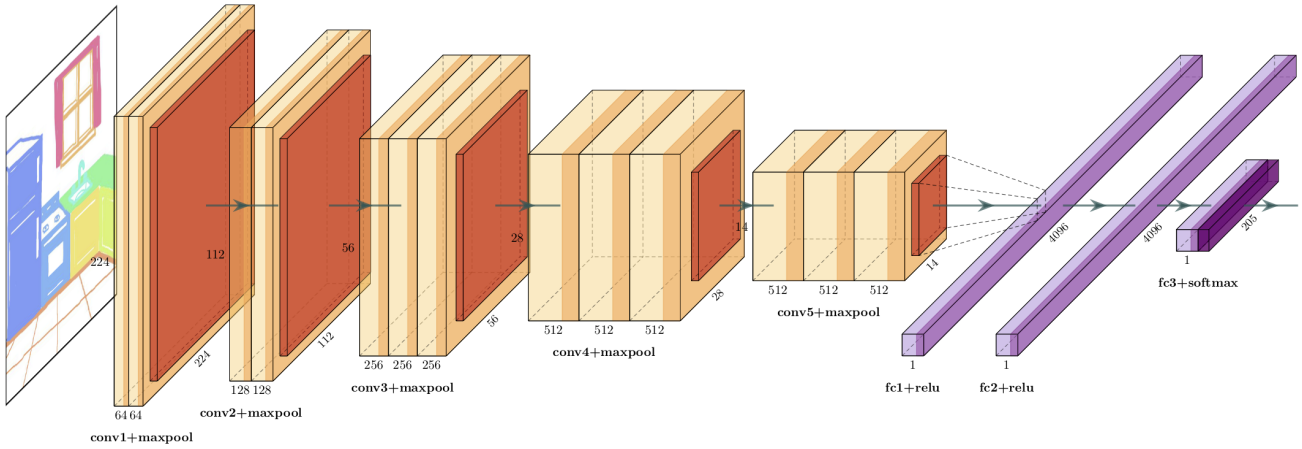


Figure 2. **VGG16 Architecture** The picture on the left shows the input of the network, consisting of an RGB image. The network consists of packages of convolution layers followed by an activation function, namely a rectified linear unit (ReLU) and finally a maxpooling layer. Those packages are repeated and followed by three fully connected layers. Convolution and maxpooling layers, shown in orange and red are frozen during training. The fully connected layers (here abbreviated with fc) make up the classifier, shown in purple. Everything prior to the classifier is frozen and only the classifier is trained. Details about the parameters, the labels and the shape of the operations can be found in appendix 4 and appendix 5.

performance-optimized hierarchical models are with neural responses in the higher visual cortex. Even going that far that deep hierarchical neural networks are beginning to be useful in the field of computational neuroscience to improve performance of sensory models (Yamins & DiCarlo, 2016).

Methods

This part describes the approach this paper takes for transferring scene representations. The goal is to learn a representation that is optimized for one of the three different modalities. For this purpose a pretrained ConvNet is modified by freezing the convolutional base and training the randomly initialized classifier. The implementation of the network is done with Keras and Tensorflow as backend. This section first introduces basics of deep learning building on the concepts explained in the previous sections. In line with this, information about the dataset and the model are given. Subsequently, the training process and relating hyperparameters are described.

Basics of Deep learning

Before deep learning, engineers needed to handcraft features specialized for narrow tasks. But representations engineered by humans were often inadequate for capturing salient semantics for a given task. By introducing convolutional filters and backpropagation (LeCun et al., 1989) for deep networks, this process was successfully automatized. These techniques are directly inspired by a biological model of simple and complex cells in visual neuroscience (Hubel & Wiesel, 1962). ConvNets and pooling layers enable the network to find relevant features or information, extract them, and obtain discriminative features from the data, called representational learning.

Mapping of inputs (I.) is the technique used in deep learning for dimensionality reduction. During all techniques used in deep learning, information is stored in a container for numbers, called a tensor. Tensors are generalizations of scalars, vectors and matrices to an arbitrary number of indices. Images and labels get equally converted into those tensors, whereby image tensors are 4D tensors (sample size, height, width and number of color channels) and labels are 1D tensors (Chollet, 2017). The color channels work with the RGB color model that can represent a broad array of colors by mixing, red, green, and blue. These tensors are used as input and output of the model. Subsequently, the goal is to find a representation through layered representation learning that is able to map the input variables via a function to the output variable. The function is based on techniques derived from calculus to perform continuous transformation. To find the function two methods are used, backpropagation and gradient descent which will be explained in detail in the training section.

Dataset

The input variables consist of natural images from the Scene205 dataset collected and curated by B. Zhou et al. (2014). An exhaustive list of the classes can be found in appendix 1. The set contains around 2.5 million images from 205 scene categories. The compressed file contains resized

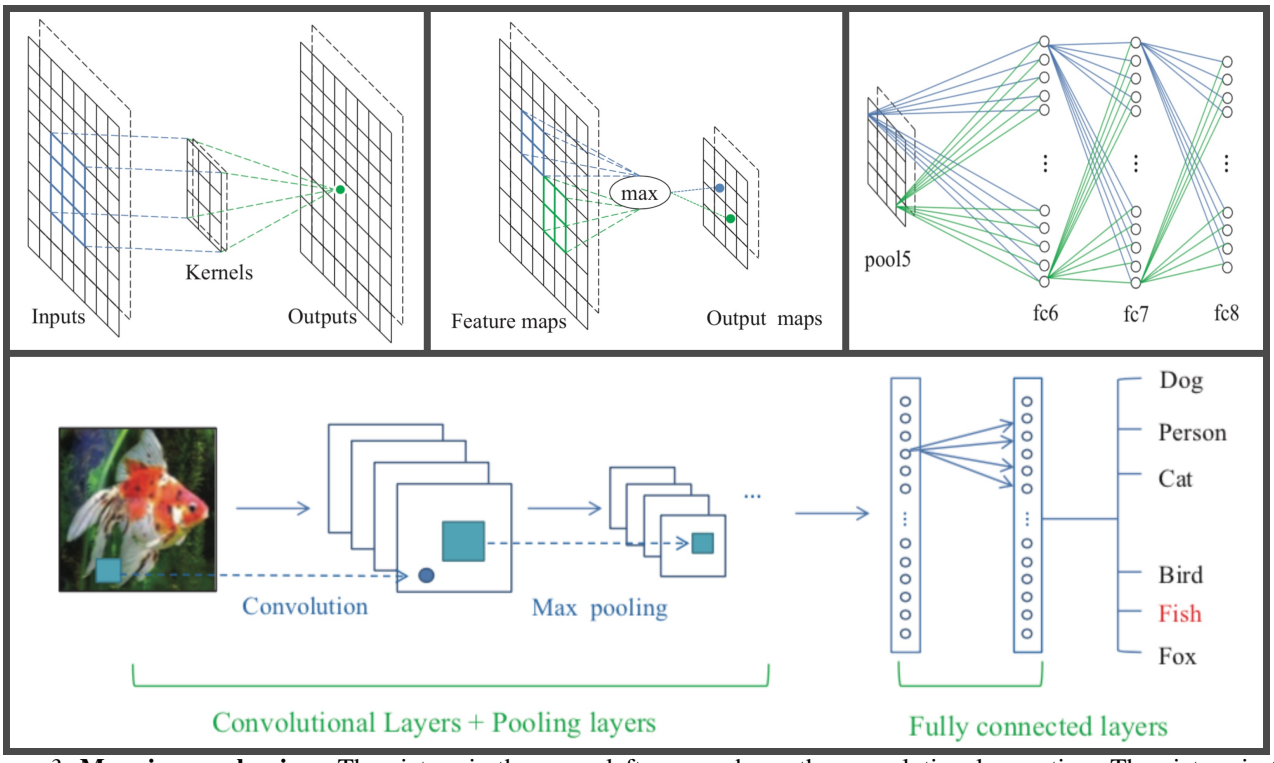


Figure 3. **Mapping mechanisms** The picture in the upper left corner shows the convolutional operation. The picture in the upper middle shows the max pooling operation and the picture on the upper right shows a schematic representation of the fully connected layers. The bottom picture summarizes the mechanisms and is a simplification of the figure 2. The pictures are taken from Yanming (2017).

256*256 images, split into a train set and a validation set of Places 205 with a size of 126GB. Due to computational limitations the number of pictures is reduced to around 80 pictures per object, so that 16,400 pictures remain. The sketch images consist of 14,830 training and 2,050 validation sketches collected through Amazon Mechanical Turk, whereby different colors indicate different objects. The clip art data includes 11,372 training and 1,954 validation clip art images downloaded from search engines. Sketches and clip art were assembled by Aytar et al. (2018). Examples for scene labels are bookstore, lobby and harbor. The same labels are used across all three domains. Representative pictures can be seen in *figure 1*.

Model

To assess the research questions in an experimental manner, a pretrained network is used instead of training a network from scratch. The image classification architecture chosen is the VGG16 assembled by Simonyan and Zisserman (2014). It is well established and has a feedforward architecture. This way the network is well equipped to map the input, the images, to the output, the labels. The VGG16 architecture is retained, consisting of 14.714.688 million parameters partioned over 13 convolutional layers. The 5 maxpooling layers and the ReLU which is applied after every convolutional layer, do not add further parameters.

Convolutions and ReLU (II.) are the core building blocks in ConvNets. Convolutional networks convert an input via a kernel function that slides over the input into an output called feature map. The convolution operation can be seen in the upper left picture of *figure 3*. It is used as an edge detector or filter and works by convoluting a kernel and an input. The filter value are not static but learned during the process. Compared to fully connected layers convolutional layers have sparse weights, this way it is unnecessary for every output unit to interact with every input unit. The kernels are smaller than the input, this way they reduce the amount of parameters (Goodfellow, Bengio, & Courville, 2016). One advantage is that weights this way can be shared which reduces the memory and computing requirements. Another advantage is that kernels are translational invariant (Chollet, 2017), meaning that a single kernel per feature map allows to detect a feature irrespective of where it is in the previous layer but the resulting activation map still encodes the location. This allows kernels to recognize an object even when its appearance varies in some way (Goodfellow et al., 2016). The next stage in a ConvNet consists of a nonlinear activation function. During this step the obtained feature maps are

passed through a non-linearity such as a rectified linear unit (ReLU). ReLU is the most popular non-linear function consisting of a rectifier $f(z) = max(z, 0)$. The rectified linear unit is zero when $x < 0$ and linear with a slope of 1 when $x > 0$, leading to an activation that is thresholded at zero. This is an advantage compared to e.g. simple linear units since it enables the network to compute nontrivial problems. ReLu was first used by Krizhevsky et al. (2012), they showed that it allows faster training of a deep supervised network without unsupervised pre-training.

Pooling (III.) is a way of sub-sampling. Average pooling for example replaces the output at a certain location with a summary of nearby inputs (Goodfellow et al., 2016). In the network a max-pooling operation is applied which returns the maximum value for a set of neighboring neurons (Y. T. Zhou & Chellappa, 1988). The maxpooling operation can be seen in the upper middle picture of *figure 3*. This form of subsampling yields the advantage of progressively reducing the size of the representation. The superiority of maxpooling layers over other pooling operations is shown in Scherer, Müller, and Behnke (2010). The organization of layers can be seen in *figure 2*. In the figure it is shown that the described operations are used several times to produce the desired output. This iterative process is well suited for mapping an RGB image to a label.

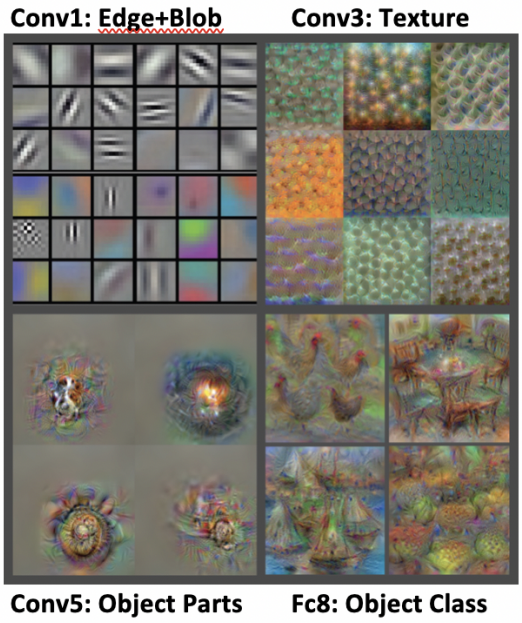


Figure 4. **Inside a convolutional network:** The figure shows representations of the AlexNet trained on the ImageNet dataset visualized by Mahendran and Vedaldi (2015).

This **hierarchy (IV.)** of consecutive convolution and maxpooling layers ensures the extraction of relevant features. This hierarchical organization allows to build representations of increasing complexity. CNNs can encode different levels of visual information, grasping the underlying representations (Bengio, Courville, & Vincent, 2013). Examples of such features can be seen in *figure 4*, showing the increasing complexity with increasing layer depth. The figure shows representations of the AlexNet trained on the ImageNet dataset visualized by Mahendran and Vedaldi (2015). Mahendran and Vedaldi (2015) also showed that the visualizations across different architectures and datasets are comparable. Lower level CNN features look like Gabor filters and are not specific to a particular dataset, unlike last-layer features. The representation progresses from local, generic feature maps in the lower levels to more-abstract concepts in the upper parts. The visualization also accentuates why a deep architecture is crucial. An architecture with a large number of layers has a representational advantage compared to shallow learners. Only the former can grasp the level of abstraction and the amount of patterns necessary for thousands of objects (Szegedy et al., 2015; Bengio et al., 2011).

As seen in *figure 2* the last part of the VGG16 consists of three **fully connected layers (V.)**. The name fully connected refers to the fact that the layers have connections to all activations in the previous layer (Goodfellow et al., 2016). The first two layers of densely connected layers integrate the information from the previous layers into a coherent whole. The last fully-connected layer holds the output, such as the class scores. The last layer is called the 'classifier' of the network since it matches the features with the actual labels (Chollet, 2017). Between the individual fully connected layers a ReLU function is applied. The last layer is followed by a softmax function that shows the obtained results in an array of 205 probability scores (summing to 1). The pretrained weights that were implemented into the VGG16 come from B. Zhou, Lapedriza, Khosla, Oliva, and Torralba (2017), and were trained on the Places365-Standard composed of 365 scene categories. This network holds the initial representation, and is a reasonable starting point.

Training

Training Set	Test Set	Name of the Results
Natural Images	Natural Images	Natural
Natural Images	Sketches	Natural-Sketch
Natural Images	Clip Art	Natural-Clip Art
Sketches	Sketches	Sketches
Sketches	Natural Images	Sketches-Natural
Sketches	Clip Art	Sketches-Clip Art
Clip Art	Clip Art	Clip Art
Clip Art	Sketches	Clip Art-Sketch
Clip Art	Natural Images	Clip Art-Natural

Table 1. Showing the different training and test combinations.

During training, the initial representation is kept constant and the classification mechanism is adjusted. This is done

with feature extraction by training the model end to end with a frozen convolutional base. The convolutional layers that learn local patterns and more abstract arrangements are assumed to hold useful representations already which would be destroyed by large weight updates. Only the last part, the fully connected layer, will be randomly initialized and retrained. To assess the question whether a network trained on natural images can classify clip art and sketches, a classifier is put on top of the convolutional base, first trained on natural images and then tested on itself and the other two domains. This procedure establishes the original baseline, which should be close to the value achieved by B. Zhou et al. (2017). Subsequently it is analyzed if better results are achieved by training the classifier on the sketches and testing on sketches afterwards. Throughout the process, the split between training set and validation set is kept. All 3 training combinations and all 9 test combinations are shown in *table 1*. There are three independent training sessions on each image set, all trained with the same learning rate. The models are tested on the test sets of their own data and consecutively on the other sets they haven't been trained on. During the whole training the convolutional base is frozen.

The image classification architecture is trained with **backpropagation and stochastic gradient descent (VI.)**. This training procedure combined gives deep learning its adaptive structure and the ability to work on different datasets. The backpropagation algorithm was proposed by Rumelhart, Hinton, and Williams (1988) and later implemented to train convolutional networks by LeCun et al. (1990). During training, the backpropagation algorithm is used to indicate which parameters need to change to compute the combination of previous layer features that best fit the constraints defined by the cost function. Backpropagation calculates the chain rule, highly efficiently, which is used to determine the derivative of the cost function. Backpropagation is needed to determine how much each weight in the network proportionally contributes to the overall error. Hence, the gradient of the cost function is needed to update the network optimally. Even with this method training takes time and is computationally expensive. For this reason only the classifier of the VGG16 will be retrained. Parts of the mathematical implementation of the backpropagation algorithm are shown in appendix 2. The loss function is categorical crossentropy, in conjunction with a softmax output activation function, since the problem is a multi-class classification with a single label per image. The loss function measures the error between output score and the desired pattern of scores. The optimizer, necessary to define how the network will be updated based on the loss function, is stochastic gradient descent (SGD). SGD uses the gradient computed during backpropagation to adjust the weights. During this process SGD uses subsets of the overall dataset (minibatches) to find a local minumum. The minimum is reached when a set of weights for all layers in the network is found that maps example inputs as good as possible to their associated targets. The step size of the SGD is defined by the learning rate (Azizpour et al., 2014). Details about the SGD algorithm can be found in appendix 3. For training a Macbook Pro with a 2,3-GHz dual-core Intel Core i5-processor was used. The learning rate was found through training the clip art training set with 5 different learning rates (ranging from 1 to $1e^{(-4)}$) for 21 epochs each. The clip art dataset was chosen for optimization because the amount of available training images was the lowest. The accuracy values achieved on the validation set were most steady for the learning rate of $lr = 1e^{(-4)}$ (0.0001). It took an overall training time of 181 hours (7.5 days) to obtain this result. Details can be obtained from appendix 6 and appendix 7. This learning rate was chosen to be the one best suited for training all data sets. The training on the actual data sets was again run for 21 epochs. The batch size was set to 32 for training and validation data. Training time per epoch was 2.3 hours on average per epoch, so taking an average 49 hours per domain.

Results and Analysis

This section summarizes the findings by first giving the chance-based baseline results, second by stating in detail the accuracy values obtained on the different test sets and third by analyzing the results.

Baseline Results

The value obtained by chance on the input data are $1/205 = 0.49\%$, this value is a bottom line. The values achieved by B. Zhou et al. (2014) on the original Places205 yield a classification accuracy of 50.2%. The Places365-VGG applied during the current study has a top-1 accuracy of 55.24% on the Places365 set (B. Zhou et al., 2017). Accuracy values for the Places665-VGG obtained on the Places205 dataset are not available.

Accuracy values

For natural images the network needed 22 epochs to reach a training accuracy of 91.81% with a maximum validation accuracy of 16.15%. For sketches 19 epochs were necessary to achieve a training accuracy of 79.59% and a validation accuracy of 18.38%. For clip art 19 epochs were necessary to achieve a training accuracy of 95.90% and a validation accuracy of 32.00%. The results are shown in *table 2*. Detailed information about the training process can be found in appendix 8. The optimal point of training was chosen before the models started to overfit. Since overfitting occurs when a model fits too well to the training set this point was reached when the validation accuracy started to decline. The accuracy of the validation set is a metric for the generalization performance of the network.

Name of the Results	Accuracy
Natural	16.150%
Natural-Sketch	1.797%
Natural-Clip Art	4.973%
Sketches	18.380%
Sketches-Natural	1.386%
Sketches-Clip Art	6.623%
Clip Art	32.000%
Clip Art-Sketch	4.083%
Clip Art-Natural	2.566%

Table 2. Showing the different training-test combinations and their accuracy values. The accuracy values for the retrained classifier are obtained on the validation set.

All models performed best on the domain they were trained on. The training accuracy of clip art increases fastest and is above 81.27% after 8 epochs. This is the point where the discrepancies between the three modalities is the largest. This could this could be due to the fact that the model already has a training accuracy of 16.14% in the first epoch. After 8 epochs, sketches have a training accuracy of 31.94% and natural images of 37.45%. The training accuracy for clip art for all epochs remains higher than the results achieved on the other two sets. The largest positive step between two consecutive epochs is between epoch 8 and epoch 9 for natural images. Here, the model gained 5.38 percentage points on the validation accuracy. The largest negative step between two consecutive epochs is between epoch 16 and epoch 17 for sketches, there the model lost 3.31 percentage points on the validation accuracy. A detailed analysis of differences between epochs can be seen in appendix 9. Both, sketches and natural images took longer to reach high accuracy values on the validation set. This might be due to the fact that the learning rate is optimized for the clip art images. It appears that both, sketches and natural images could have been trained with a higher learning rate or more epochs. The validation accuracy achieved on the clip art set is nearly double the accuracy achieved on the natural image set, showing a significant difference. This is especially interesting since the network was pre-trained on this domain and because the natural images did not come close to the values achieved by B. Zhou et al. (2017).

Another goal of the paper was to to learn a multi-modal representation of different domains. Therefore, the obtained networks were consecutively tested on the datasets they were not trained on. The testing on the other modalities was done on every training and validation set and afterwards combined by calculating the weighted average in the following way:

$$\frac{acc_{(train)} * n_{(train)} + acc_{(val)} * n_{(val)}}{n_{(train)} + n_{(val)}} = accuracy$$

Where *acc* is referring to the accuracy, *train* to training data, *val* to validation data and *n* is the number of images. Using this formula the classifier trained on natural images achieves an accuracy value of 1.80% on the sketch images and a value of 4.97% on the clip art data. This answers the questions statet at the beginning, how well the chosen image classification architecture performs on classifying clip art and drawn images. The classifier trained on the sketch dataset achieves an accuracy value of 1.39% on the natural images and a value of 6.62% on the clip art set. The image classification architecture trained on clip art images yields an accuracy score of 4.08% on the sketches and a value of 2.57% on the natural image set. The results are shown in *table 2*. The best overall values were achieved with the classifier trained on clip art. All values mentioned in this section were above the chance level of 0.49%.

Analysis

This part uses the obtained results to answer the research questions stated at the beginning. A comparison between prior hypotheses and the obtained accuracy values is made. Further, an interpretation of the results is stated, giving possible explanations for the unexpected results.

Is the pretrained ConvNet reusable for classifying clip art and sketches? The best results were obtained on clip art. With an accuracy of 32.00% the pretrained ConvNet holds enough prior knowledge to achieve such a mixed result. For sketches the frozen convolutional base was also able to provide enough information to reach an accuracy close to classifying one out of five of the images correctly. This targets the second question whether the features extracted from natural images are reusable for classifying clip art and sketches. However, the result is opposing the research hypothesis and prior expectations. It was assumed that since the convolutional base and the target domain are derived from the same dataset they were expected to perform best. One reason why this might not be the case is that the ConvNet was trained on the Places365 and the classifier was trained on the Places205 dataset. The main limitation do not seem to be the difference in the input distribution but instead the transferability is limited by the diversity within domains. The diversity within a domain is comparable to the width of the input distribution. For clip art and sketches it seems like the main assumption underlying domain adaptation is partly violated due to dissimilarity of features across the domains. Groen, Ghebreab, Prins, Lamme, and Scholte (2013) already proposed the explanation that man-made and natural scenes are fundamentally different in scene perception. But due to the low values achieved on the natural images it could even be argued that the distance between the two domains is not that far. However, it remains questionable how good the results would have been if a classifier was trained on a for example randomly initialized network. Such results as comparison could further illustrate the role of prior representations. While performing the analysis, an additional question

emerged, namely whether features learned on one domain generalize well. It is assumed that a model that generalizes well on other domains hold multiple representations simultaneously.

Do features learned on one domain generalize well? The results for multimodality all remain below 7.00%, which is still above chance level but essential lower than the values achieved on the modality they were trained on. The domain that generalizes best is clip art. The clip art data performed with the highest accuracy on the other two datasets and also had a reasonable performance itself. Hence, clip art holds the best multimodal representation comparing the 3 classifiers. Regarding the accuracy values it can be concluded that the input distributions from sketches and clip art are closer than the input distribution of natural images to sketches or clip art. Not only do both models perform best on each other (clip art on sketches 4.08%, sketches on clip art 6.62%), but also better than natural images (natural on sketches 1.80%, natural on clip art 4.97%).

Overall, multiple representations within the same network seem possible but only in a narrow range. These results possibility could be increased by training the model in a truly multimodal fashion. The current classifiers were not further trained to be explicitly multimodal. To a certain amount the model was still able to leverage the obtained knowledge across different domains. The trade-off between being multimodal and being optimally suited for one domain could not be further analyzed in this context.

Finally, the task of classifying places with limited amounts of data is difficult for several reasons. Since the labels of classes rely on underlying human concepts there is no guarantee that each class is equally diverse or rich. Concepts can go beyond physical appearance of objects. A kitchen for example is not only characterized by the objects in it, but also through the activities an individual does there (e.g. cooking, eating). This might be one of the reasons why natural images with limited amounts of data were classified worse than clip art. Clip art is more prone to capture stereotypical features of the class only, this possibly reduces the richness and makes it easier to generalize to data not seen before. For natural images the bias towards stereotypical images is not existing, this is an explanation why the limited amount of 16.400 images might not have been enough to build up a reasonable representation of the domain.

Discussion

The aim of the present study was to determine whether learned representations are transferable from natural images to clip art or drawings. Further, the ability of networks to hold multiple representations simultaneously was studied. The results can be summarized as follows. First, the overall transferability of learned features is not only limited by the distance but also by the diversity of the input distribution. Second, the representations of clip art achieve the highest results in the current study. Third, concerning the possibility of multi-modal representations it was shown that such representations perform relatively poor. The current study raises questions concerning the reusability of fully connected layers when not properly assessing the optimal learning rate beforehand. Especially the bad performance of the network trained with natural images needs further investigation.

Limitations A major drawback of the current study is how poorly the dataset is curated. The datasets were generated by humans but not validated by humans afterwards. In some cases it remains doubtful if the images could be classified correctly by humans. This makes it eminently hard to classify some images correctly. The problem was especially evident in the sketches images, which was shown by the low overall accuracy. An example can be seen in figure 5. Another problem was the availability of unexpected additional information in the clip art dataset, leading to an overestimation of predictability. The additional information was provided by watermarks on the pictures. It was not further tested to what extend the watermarks were distributed equally across classes and how strong their influence on the final results was. The limited amount of available training data worsens this problem. Due to limited computational resources, training the network end-to-end for this project was out of scope. The values obtained during training the network end-to-end would have been a good value to compare the results obtained in this study to. It is for example possible that even with training the model end-to-end the results for sketches remain close to the results achieved in this study. This performance might therefore not be due to the poor transferability of features but due to the inability of convolutional networks to grasp the right features within the domain. Another possible issue is the diversity of classes. Even within a single class there is often a broadly defined range of what a class consists of. For example the class 'shopfront' could be further subdivided. Broadly defined classes make it harder to find a unified representation of the class.

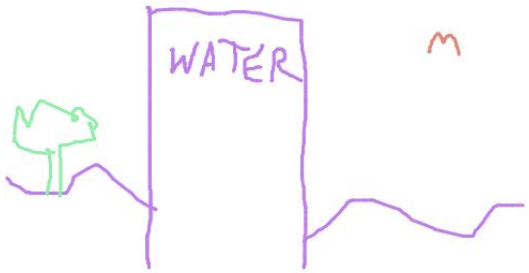


Figure 5. **Can you recognize the class?** An example of a 'tower' from the sketches dataset.

Future directions A possible solution for the 'subdivision of classes problem' is to increase the richness of the labels.

By applying this approach, the high within class diversity and individual items in a class can be classified additionally to the scene as a whole. Next, only the sum of the items is used to classify the scene. A subsequent comparison to the current study might give clarification which approach is better suited for classifying scene images.

Improve the baseline As already mentioned in the limitations, determining an accuracy baseline with a network that was trained end-to-end yields potential information. If the values achieved during the end-to-end training are not above the values achieved in this study, the input distributions might not be that different.

Between domains Furthermore, an improvement of the dataset's quality might significantly affect the results. By deleting or replacing images that are not classifiable by humans, a quality improvement could be achieved. Regularization methods might increase the values significantly. Such methods make modifications to the learning algorithm or the network such that the model generalizes better (Goodfellow et al., 2016). However, the questions of which representations are better transferable than others and why remains. Nevertheless, absolute or relative differences in accuracy gains between domains yield potential information necessary for further analysis.

Between classes A next step towards getting a more complete picture about the transferability of features is to look if large differences between classes exist. If such differences exist it is necessary to explore how the differences can be justified. If accuracy values obtained for a bathroom compared to those of an office room differ notably, further examination should be considered. In a similar investigation, Donahue et al. (2013) found that feature trained on indoor and outdoor scenes differ significantly.

Colours Comparable to studying between classes differences is an investigation of the effects of color on transferabiliy. Sketch images are, compared to the other two image sets, less colorful and less nuanced in their color grading and richness. The use of black and white color channels shed light on the question of how big the role of different color grading is for the transferability. For the sketch domain only between four and six colours were used. It is expected that the validation accuracy of this domain suffers less than the other domains.

Multi-modal representations To test whether multi-modal representations are possible to exist within the same network, the network can be trained with multiple domains simultaneously. The mixing of different domains can lead to the best possible multi-modal network and to a strong alignment across domains. It remains questionable if the overall representation of two domains can coexist or if there is always a trade-off situation.

Input distribution In the future, it will be important to find reliable techniques that are able assess the distance between two domains to see whether transfer learning is possible. The measurement of distances between input distributions is inherently difficult because a method to boil down the characteristics of an input distribution has not been found yet. This issue leads to the question how to optimally set the hyperparmaters for a network. Often only the consecutive testing of different settings discloses the best option and defines the model that is able to grasp statistical regularities. Since recently these hyperparameters were set by humans, but the trend might shift to quantifying input distributions beforehand during the next years. This would decrease training time and make for example convolutional neural networks available to a broader spectrum of users.

Acknowledgments

I thank Alexander Kroner for helpful discussions and guidance, as well as funding from the Maastricht Research Based Learning (MaRBLe) program. I would also like to thank Salomé-Marie Porten for providing me with continuous encouragement.

References

Aytar, Y., Castrejon, L., Vondrick, C., Pirsiavash, H., & Torralba, A. (2018). Cross-modal scene networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*, 2303-2314. doi: 10.1109/TPAMI.2017.2753232

Aytar, Y., & Zisserman, A. (2011, Nov). Tabula rasa: Model transfer for object category detection. , 2252-2259. doi: 10.1109/ICCV.2011.6126504

Azizpour, H., Razavian, A. S., Sullivan, J., Maki, A., & Carlsson, S. (2014). From generic to specific deep representations for visual recognition. *CoRR*, *abs/1406.5774*. doi: 10.1109/CVPRW.2015.7301270

Bengio, Y., Bastien, F., Bergeron, A., Boulanger-Lewandowski, N., Breuel, T., Chherawala, Y., ... Sicard, G. (2011, 11-13 Apr). Deep learners benefit more from out-of-distribution examples. , *15*, 164–172.

Bengio, Y., Courville, A., & Vincent, P. (2013, Aug). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798-1828. doi: 10.1109/TPAMI.2013.50

Bishop, C. M. (2006). *Pattern recognition and machine learning (information science and statistics)*. Berlin, Heidelberg: Springer-Verlag.

Chollet, F. (2017). *Deep learning with python*. Greenwich, CT, USA: Manning Publications Co.

Coutinho, E., Deng, J., & Schuller, B. (2014, July). Transfer learning emotion manifestation across music and speech. In *2014 international joint conference on neural networks (ijcnn)* (p. 3592-3598). doi: 10.1109/IJCNN.2014.6889814

Csurka, G. (Ed.). (2017). *Domain adaptation in computer vision applications*. Springer.

DiCarlo, J. J., & Cox, D. D. (2007). Untangling invariant object recognition. *Trends in Cognitive Sciences*, *11*, 333-341. doi: 10.1016/j.tics.2007.06.010

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2013). Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, *abs/1310.1531*.

Fu, Y., Xiang, T., Jiang, Y., Xue, X., Sigal, L., & Gong, S. (2017). Recent advances in zero-shot recognition. *CoRR*, *abs/1710.04837*.

Ganin, Y., & Lempitsky, V. (2014, September). Unsupervised Domain Adaptation by Backpropagation. *arXiv e-prints*, arXiv:1409.7495.

Geirhos, R., Janssen, D. H. J., Schütt, H. H., Rauber, J., Bethge, M., & Wichmann, F. A. (2017). Comparing deep neural networks against humans: object recognition when the signal gets weaker. *CoRR*, *abs/1706.06969*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. (`http://www.deeplearningbook.org`)

Groen, I. I., Ghebreab, S., Prins, H., Lamme, V. A., & Scholte, H. S. (2013). From image statistics to scene gist: Evoked neural activity reveals transition from low-level natural image structure to scene category. *The Journal of Neuroscience*, *33*(48), 18814-18824. doi: 10.1523/jneurosci.3128-13.2013

Hassabis, D., Kumaran, D., Summerfield, C., & Botvinick, M. (2017). Neuroscience-inspired artificial intelligence. *Neuron Review*, *45*, 245-258. doi: 10.1016/j.neuron.2017.06.011

Hawkins, J., Lewis, M., Klukas, M., Purdy, S., & Ahmad, S. (2018). A framework for intelligence and cortical function based on grid cells in the neocortex. *bioRxiv 442418*. doi: //doi.org/10.1101/442418

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of physiology*, *160*, 106-154. doi: 10.1113/jphysiol.1962.sp006837

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th international conference on neural information processing systems - volume 1* (pp. 1097–1105). USA: Curran Associates Inc.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2016). Building machines that learn and think like people. *CoRR*, *abs/1604.00289*.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989, Dec). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541-551. doi: 10.1162/neco.1989.1.4.541

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., & Jackel, L. D. (1990). *Handwritten digit recognition with a back-propagation network* (D. S. Touretzky, Ed.). Morgan-Kaufmann.

Luo, Y., Zheng, L., Guan, T., Yu, J., & Yang, Y. (2018). Taking A closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. *CoRR*, *abs/1809.09478*.

Mahendran, A., & Vedaldi, A. (2015). Understanding deep image representations by inverting them. In *Ieee conference on computer vision and pattern recognition.* doi: 10.1109/CVPR.2015.7299155

Markov, N. T., Vezoli, J., Chameau, P., Falchier, A., Quilodran, R., Huissoud, C., ... Kennedy, H. (2014). Anatomy of hierarchy: Feedforward and feedback pathways in macaque visual cortex. *Journal of Comparative Neurology*, *522*(1), 225-259. doi: 10.1002/cne.23458

Milne, A., Wilson, B., & Christiansen, M. (2018). Structured sequence learning across sensory modalities in humans and non-human primates. *Current Opinion in Behavioral Sciences*, *21*, 39-48.

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., & Ng, A. Y. (2011). Multimodal deep learning. In *Proceedings of the 28th international conference on international conference on machine learning* (pp. 689–696). USA: Omnipress.

Pan, S. J., & Yang, Q. (2010, October). A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, *22*(10), 1345-1359. doi: 10.1109/TKDE.2009.191

Riesenhuber, M., & Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, *2*, 1019–1025.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. doi: 10.1037/h0042519

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Neurocomputing: Foundations of research. In J. A. Anderson & E. Rosenfeld (Eds.), (pp. 696–699). Cambridge, MA, USA: MIT Press.

Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *Proceedings of the 20th international conference on artificial neural networks: Part iii* (p. 92-101). Berlin, Heidelberg: Springer-Verlag. doi: $10.1007/978 - 3 - 642 - 15825 - 4_10$

Sejnowski, T. J. (2018). *The deep learning revolution*. MIT Press.

Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M., & Poggio, T. (2007, March). Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(3), 411-426. doi: 10.1109/TPAMI.2007.56

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*.

Socher, R., Ganjoo, M., Sridhar, H., Bastani, O., Manning, C. D., & Ng, A. Y. (2013). Zero-shot learning through cross-modal transfer. *CoRR*, *abs/1301.3666*.

Soekhoe, D., van der Putten, P., & Plaat, A. (2016). On the impact of data set size in transfer learning using deep neural networks. In *Ida.*

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2015). *Going deeper with convolutions*. doi: 10.1109/CVPR.2015.7298594

Wang, W., Zheng, V. W., Yu, H., & Miao, C. (2019, January). A survey of zero-shot learning: Settings, methods, and applications. *ACM Trans. Intell. Syst. Technol.*, *10*(2), 13:1–13:37. doi: 10.1145/3293318

Yamins, D. L. K., & DiCarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, *19*, 356-365.

Yamins, D. L. K., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D. A., & DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences of the United States of America*, *111 23*, 8619-24.

Yanming, G. (2017). *Deep learning for visual understanding* (Unpublished doctoral dissertation). ASCI graduate school TU Delf.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *CoRR*, *abs/1411.1792*.

Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., & Torralba, A. (2017). Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., & Oliva, A. (2014). Learning deep features for scene recognition using places database. In *Proceedings of the 27th international conference on neural information processing systems - volume 1* (p. 487-495). Cambridge, MA, USA: MIT Press.

Zhou, Y. T., & Chellappa, R. (1988). Computation of optical flow using a neural network. *IEEE 1988 International Conference on Neural Networks*, 71-78 vol.2.

Appendix

Appendix 1 - Classes

All 205 classes in alphabetical order: abbey, airport, terminal, alley, amphitheatre, amusement park, aquarium, aqueduct, arch, art gallery, art studio, assembly line, attic, auditorium, apartment building outdoor, badlands, ballroom, bamboo forest, banquet hall, bar, baseball field, basement, basilica, bayou, beauty salon, bedroom, boardwalk, boat deck, bookstore, botanical garden, bowling alley, boxing ring, bridge, building facade, bus interior, butchers shop, butte, bakery shop, cafeteria, campsite, candy store, canyon, castle, cemetery, chalet, classroom, closet, clothing store, coast, cockpit, coffee shop, conference center, conference room, construction site, corn field, corridor, cottage garden, courthouse, courtyard, creek, crevasse, crosswalk, cathedral outdoor, church outdoor, dam, dining room, dock, dorm room, driveway, desert sand, desert vegetation, dinette home, doorway outdoor, engine room, excavation, fairway, fire escape, fire station, food court, forest path, forest road, formal garden, fountain, field cultivated, field wild, galley, game room, garbage dump, gas station, gift shop, golf course, harbour, herb garden, highway, home office, hospital, hospital room, hot spring, hotel room, hotel outdoor, ice cream parlor, iceberg, igloo, islet, ice skating rink outdoor, inn outdoor, jail cell, kasbah, kindergarden classroom, kitchen, kitchenette, laundromat, lighthouse, living room, lobby, locker room, mansion, marsh, martial arts gym, mausoleum, medina, motel, mountain, mountain snowy, music studio, market outdoor, monastery outdoor, museum indoor, nursery, ocean, office, office building, orchard, pagoda, palace, pantry, parking lot, parlor, pasture, patio, pavilion, phone booth, picnic area, playground, plaza, pond, pulpit, racecourse, raft, railroad track, rainforest, reception, residential neighborhood, restaurant, restaurant kitchen, restaurant patio, rice paddy, river, rock arch, rope bridge, ruin, runway, sandbar, schoolhouse, sea cliff, shed, shoe shop, shopfront, shower, ski resort, ski slope, sky, skyscraper, slum, snowfield, staircase, supermarket, swamp, stadium baseball, stadium football, stage indoor, subway station platform, swimming pool outdoor, television studio, topiary garden, tower, train railway, tree farm, trench, temple east asia, temple south asia, track outdoor, train station platform, underwater coral reef, valley, vegetable garden, veranda, viaduct, volcano, waiting room, water tower, watering hole, wheat field, wind farm, windmill, yard.

Appendix 2 - The backpropagation algorithm The **cost function** is given by:

$$J(\Theta)$$

Whereby Θ is the hypothesis or simply called the model. The model is defined by the configuration of weights it holds. The cost function is a generalization of a logistic regression function. We want to optimize by $\min_{\Theta} J(\Theta)$ by taking the partial derivative. The resulting gradient in a general form is given by:

$$g = (\frac{\partial}{\partial \Theta_i^l}) * J(\Theta)$$

Whereby i stands for the index of the parameter and l for the layer. The hypothesis, which is also the output of the network is denoted by

$$h_\Theta(x)$$

.

To compute the hypothesis forward propagation is used. This is done by computing all activation values for all neurons in the neural network. The first layer activation equals the input $x^{(i)}$, defined as:

$$a^{(1)} = x^{(i)}$$

The last layer activation is shown here:

$$a^{(L)} = h_\Theta(x) = g(z^{(L)})$$

Whereby L stands for the total number of layers in the network and a^L stands for the activation values of the L layer. The term $g(z^{(L)})$ is the activity of the last layer after applying an activation function. The activation is derived from the previous layer activation taking the dot product of the hypothesis. In detail it is defined as:

$$z^{(L)} = \Theta^{(L-1)} * a^{(L-1)}$$

Backpropagation is an optimization function for updating the parameters given by the cost function $J(\Theta)$. We want to minimize the error this cost function produces and through that increase the accuracy of our model. To achieve this goal it is necessary to compute the cost function and partial derivative of the cost function. The error of node j in layer l is denoted with:

$$\delta_j^{(l)}$$

For each node in each layer it is now important to calculate the error term. For the last layer this is given by:

$$\delta_j^{(L)} = a_j^{(L)} - y_j$$

Whereby y_j stands for the actually observed value. $\delta_j^{(L)}$ (last layer nodes) are also the difference between what is the output of the hypothesis and the true value. The error values of the previous layers are not immediately obvious and need to be calculated using the chain rule. The error term of the previous layer $(L-1)$ is given by:

$$\delta^{(L-1)} = (\Theta^{(L-1)})^T \delta^{(L)} .* g'(z^{(L-1)})$$

where $g'(z^{(L-1)}) = a^{(L-1)} .* (1 - a^{(L-1)})$ is referring to the derivative of the activation function, T stands for transpose and $.*$ stands for element-wise multiplication. The error is calculated for all nodes in all layers through this iterative process, except for the first one since the first layer holds the input parameters that we don't want to change. The error is during stochastic gradient descent used to update the parameters accordingly. Without further regularization it then could be shown for all layers that:

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} * J(\Theta) = a_j^{(l)} * \delta_i^{(l+1)}$$

Algorithmic implementation Given a training set $(x^{(1)}, y^{(1)}), ..., (x^{(m)}, y^{(m)})$ and setting $\Delta_{ij}^{(l)} = 0$ for all l, i, j. What makes Δ so important is that Δ can be used to calculate $\frac{\partial}{\partial \Theta_{ij}^{(l)}} * J(\Theta)$. This is done in the following steps by looping through the training set. For $i = 1$ to m set $a^{(1)} = x^{(i)}$. Performing the forward propagation to compute $a^{(1)}$ for $l = 2, 3, ..., L$ using $y^{(i)}$ (the label), compute $\delta^{(i)} = a^{(L)} - y^{(i)}$. After computing $\delta^{(L)}, \delta^{(L-1)}, ..., \delta^{(2)}$ we can use them to compute:

$$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} * \delta_i^{(l+1)}$$

We are now able to compute:

$$D_{ij}^{(l)} := \frac{1}{m} * \Delta_{ij}^{(l)}$$

Finally:

$$D_{ij}^{(l)} := \frac{\partial}{\partial \Theta_{ij}^{(l)}} * J(\Theta)$$

Appendix 3 - Stochastic Gradient Descent Updating the parameters is done with the stochastic gradient descent algorithm. After computing the error terms with the backpropagation algorithm we can use those to update the network with stochastic gradient descent. Stochastic gradient descent (SGD) is used instead of gradient descent. SGD is less computationally expensive to compute the gradient since it doesn't look at all training examples in every single iteration. Batch gradient descent consists of repeating:

$$\Theta_j := \Theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

for every $j = 0, ..., n$. Whereby a is the learning rate, m is the amount of training examples and the other parameters are the same as in Appendix 1. In comparison SGD defines the cost function differently:

$$cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2} (h_\Theta(x^{(i)}) - y^{(i)})^{(2)}$$

resulting in the following cost function:

$$J_\theta = \frac{1}{m} \sum_{i=1}^{m} cost(\theta, (x^{(i)}, y^{(i)}))$$

Algorithmic implementation The following steps are executed during SGD:
First, randomly shuffle the data set.
Second, repeat $i = 1, .., m$ for:

$$\theta_j = \theta_j - \alpha (h_\theta(x^{(i)} - y^{(i)}) x_j^{(i)}$$

for all values of $j = 1, .., m$. Which is the actual updating of the parameters. What SGD is doing during this process is looking at an small amount of training examples and fit the parameters on them. It does not sum over all training examples.

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
None
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 205)               839885
=================================================================
Total params: 135,100,429
Trainable params: 135,100,429
Non-trainable params: 0
```

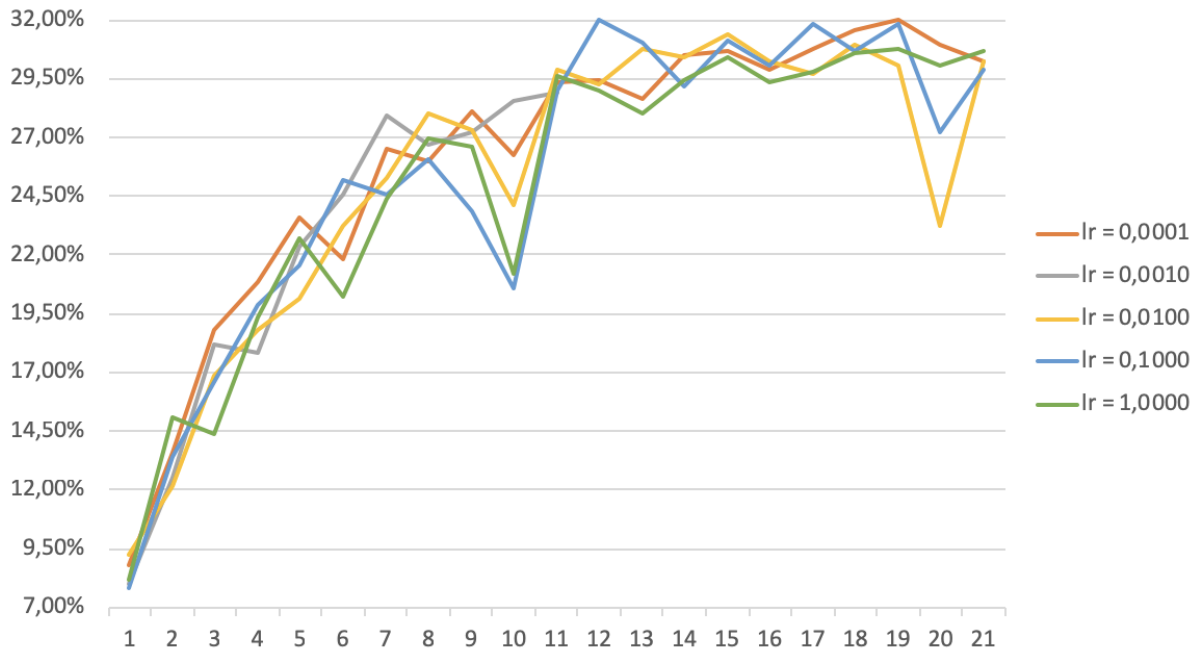
Appendix 4 - Number of Parameters The figure shows in detail the type and number of layers. Also showing the overall number of parameters before freezing, 135.100.429 trainable parameters.

```
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 205)               839885
=================================================================
Total params: 135,100,429
Trainable params: 120,385,741
Non-trainable params: 14,714,688
```

Appendix 5 - Number of Parameters Figure is showing the number of parameters after freezing, 14.714.688 trainable parameters remain and 120.385.741 are non-trainable.

	lr=1e-4		lr=1e-3		lr=1e-2		lr=1e-1		lr=1	
Learning rate	**0,0001**		**0,0010**		**0,0100**		**0,1000**		**1,0000**	
	acc	val_acc	acc	val_acc	acc	val_acc	acc	val_acc	acc	val_acc
epoch 1	16,14%	8,76%	16,29%	7,99%	15,60%	9,27%	15,36%	7,79%	15,48%	8,200%
epoch 2	32,13%	13,58%	32,34%	12,54%	31,91%	12,12%	32,79%	13,42%	32,53%	15,09%
epoch 3	44,50%	18,83%	44,20%	18,21%	43,75%	16,86%	44,07%	16,60%	44,19%	14,41%
epoch 4	53,21%	20,86%	54,20%	17,85%	53,63%	18,83%	53,26%	19,88%	53,17%	19,30%
epoch 5	61,32%	23,57%	61,89%	22,32%	62,25%	20,14%	62,30%	21,54%	62,05%	22,68%
epoch 6	69,54%	21,80%	69,52%	24,61%	70,05%	23,26%	69,43%	25,18%	69,10%	20,24%
epoch 7	75,96%	26,53%	75,94%	27,94%	76,41%	25,29%	76,06%	24,56%	76,05%	24,40%
epoch 8	81,27%	25,96%	81,80%	26,69%	81,73%	28,04%	81,58%	26,07%	81,89%	26,95%
epoch 9	86,03%	28,15%	86,31%	27,21%	86,34%	27,32%	85,99%	23,88%	86,41%	26,59%
epoch 10	89,92%	26,27%	89,78%	28,51%	89,58%	24,09%	89,60%	20,60%	89,60%	21,23%
epoch 11	91,95%	29,34%	91,87%	28,93%	91,78%	29,86%	92,11%	28,98%	92,24%	29,60%
epoch 12	93,28%	29,40%	93,27%	30,33%	93,41%	29,24%	93,10%	**32,05%**	93,44%	28,98%
epoch 13	94,15%	28,67%	94,01%	29,60%	93,93%	30,80%	94,08%	31,06%	94,04%	28,04%
epoch 14	94,87%	30,49%	94,82%	30,54%	94,78%	30,44%	94,70%	29,14%	94,82%	29,40%
epoch 15	95,06%	30,65%	95,24%	31,01%	95,15%	**31,43%**	95,15%	31,11%	95,01%	30,43%
epoch 16	95,24%	29,86%	95,26%	29,29%	95,10%	30,28%	95,66%	30,02%	95,43%	29,34%
epoch 17	95,77%	30,80%	95,64%	29,50%	95,57%	29,71%	95,49%	31,79%	95,79%	29,76%
epoch 18	95,69%	31,58%	95,53%	**31,11%**	95,70%	30,91%	95,82%	30,70%	95,68%	30,59%
epoch 19	95,90%	**32,00%**	95,62%	31,06%	95,74%	30,02%	95,85%	31,84%	95,78%	**30,80%**
epoch 20	95,84%	30,96%	95,94%	26,59%	95,71%	23,20%	95,78%	27,21%	95,71%	30,07%
epoch 21	95,90%	30,28%	95,95%	30,85%	95,84%	30,28%	95,55%	29,86%	96,03%	30,70%

Appendix 6 - Finding the optimal learning rate The table shows the process of finding the optimal learning rate. Showing the learning rates 0,0001 (1e-4), 0,0010 (1e-3), 0,01000 (1-e2), 0,1000 (1-e1), 1. The highest accuracy, 32,05% was obtained with a learning rate of 0,1. The most stable accuracy was achieved with a learning rate of 0,0001. This was the reason why this learning rate was chosen, especially since the top performance was 32,00%.



Appendix 7 - Finding the optimal learning rate Plotting of different learning ratesShowing the learning rates 0,0001 (1e-4), 0,0010 (1e-3), 0,01000 (1-e2), 0,1000 (1-e1), 1. The drop of all different learning rates around the 10 epoch is something that could not be explained within the context of this work.

	Clip Art		Sketches		Natural Images	
	acc	val_acc	acc	val_acc	acc	val_acc
epoch 1	16,14%	8,76%	1,68%	1,90%	2,30%	2,25%
epoch 2	32,13%	13,58%	5,98%	5,05%	7,43%	6,39%
epoch 3	44,50%	18,83%	11,02%	6,99%	13,10%	7,43%
epoch 4	53,21%	20,86%	15,35%	6,10%	18,10%	8,87%
epoch 5	61,32%	23,57%	19,42%	6,00%	23,31%	8,72%
epoch 6	69,54%	21,80%	24,03%	5,25%	27,86%	9,02%
epoch 7	75,96%	26,53%	27,53%	6,49%	32,63%	10,36%
epoch 8	81,27%	25,96%	31,94%	9,67%	37,45%	8,25%
epoch 9	86,03%	28,15%	35,51%	15,01%	42,90%	13,63%
epoch 10	89,92%	26,27%	40,69%	14,87%	48,60%	12,54%
epoch 11	91,95%	29,34%	44,46%	13,92%	54,63%	12,24%
epoch 12	93,28%	29,40%	49,07%	15,51%	59,43%	13,97%
epoch 13	94,15%	28,67%	53,36%	16,40%	65,95%	14,77%
epoch 14	94,87%	30,49%	58,16%	16,20%	70,05%	14,42%
epoch 15	95,06%	30,65%	63,16%	16,20%	74,99%	15,31%
epoch 16	95,24%	29,86%	67,44%	18,24%	78,67%	15,06%
epoch 17	95,77%	30,80%	72,17%	14,93%	81,79%	15,16%
epoch 18	95,69%	31,58%	76,14%	17,94%	85,60%	15,56%
epoch 19	95,90%	**32,00%**	79,56%	**18,38%**	87,42%	14,37%
epoch 20	95,84%	30,96%	83,93%	17,44%	89,24%	15,96%
epoch 21	95,90%	30,28%	87,04%	18,33%	90,51%	13,73%
epoch 22					91,81%	**16,15%**
epoch 23					92,69%	15,86%
epoch 24					93,65%	15,01%
epoch 25					93,82%	16,01%
epoch 26					94,76%	15,41%
epoch 27					94,75%	15,91%
epoch 28					95,27%	15,76%

Appendix 8 - Overview training classifiers Number of epochs trained and the training and validation accuracy

	Clip Art		Sketches		Natural Images	
	acc	val_acc	acc	val_acc	acc	val_acc
epoch 2 - epoch 1	15,99%	4,82%	4,30%	3,15%	5,13%	4,14%
epoch 3 - epoch 2	12,37%	5,25%	5,04%	1,94%	5,67%	1,04%
epoch 4 - epoch 3	8,71%	2,03%	4,33%	-0,89%	5,00%	1,44%
epoch 5 - epoch 4	8,11%	2,71%	4,07%	-0,10%	5,21%	-0,15%
epoch 6 - epoch 5	8,22%	-1,77%	4,61%	-0,75%	4,55%	0,30%
epoch 7 - epoch 6	6,42%	4,73%	3,50%	1,24%	4,77%	1,34%
epoch 8 - epoch 7	5,31%	-0,57%	4,41%	3,18%	4,82%	-2,11%
epoch 9 - epoch 8	4,76%	2,19%	3,57%	5,34%	5,45%	5,38%
epoch 10 - epoch 9	3,89%	-1,88%	5,18%	-0,14%	5,70%	-1,09%
epoch 11 - epoch 10	2,03%	3,07%	3,77%	-0,95%	6,03%	-0,30%
epoch 12 - epoch 11	1,33%	0,06%	4,61%	1,59%	4,80%	1,73%
epoch 13 - epoch 12	0,87%	-0,73%	4,29%	0,89%	6,52%	0,80%
epoch 14 - epoch 13	0,72%	1,82%	4,80%	-0,20%	4,10%	-0,35%
epoch 15 - epoch 14	0,19%	0,16%	5,00%	0,00%	4,94%	0,89%
epoch 16 - epoch 15	0,18%	-0,79%	4,28%	2,04%	3,68%	-0,25%
epoch 17 - epoch 16	0,53%	0,94%	4,73%	-3,31%	3,12%	0,10%
epoch 18 - epoch 17	-0,08%	0,78%	3,97%	3,01%	3,81%	0,40%
epoch 19 - epoch 18	0,21%	0,42%	3,42%	0,44%	1,82%	-1,19%
epoch 20 - epoch 19	-0,06%	-1,04%	4,37%	-0,94%	1,82%	1,59%
epoch 21 - epoch 20	0,06%	-0,68%	3,11%	0,89%	1,27%	-2,23%

Appendix 9 - Overview of changes during training Showing the differences of training results